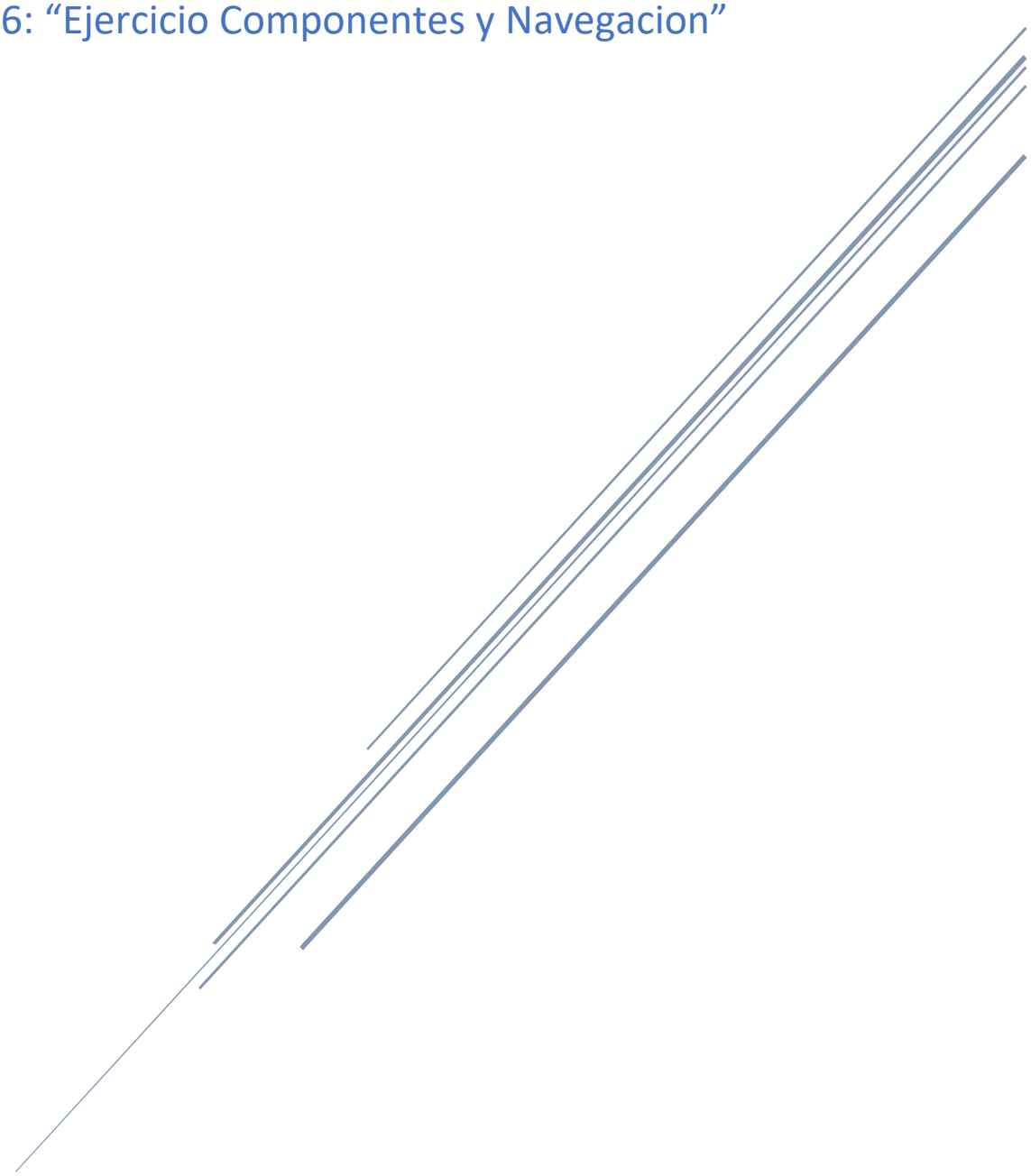
upna

Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

APP GAZTAROA

Commit 06: “Ejercicio Componentes y Navegacion”



Naiara Platero

Ejercicio Componentes y Navegación

En este ejercicio crearemos dos nuevos componentes y los integraremos en la aplicación. En concreto, se trata de un componente para mostrar una página de contacto y un segundo componente para comentar brevemente la historia del club de montaña y el tipo de actividades que realiza.

1. Componente Contacto

En la carpeta común se crea un array que se guarda en una constante con nombre *CONTACTOS.* Para realizar los saltos de línea se utiliza \n.

```
export const CONTACTOS =
    [
    {
        id: 0,
        nombre: 'Información de contacto',
        destacado: true,
        descripcion: 'Kaixo Mendizale!'+"\n"+
        ' '+"\n"+
        'Si quieres participar en las salidas de montaña que organizamos o quieres hacerte soci@ de Gaztaroa, puedes
        ' '+"\n"+
        'Para lo que quieras, estamos a tu disposición!'+"\n"+
        ' '+"\n"
        +' Tel: +34 948 277151'+"\n"+
        ' '+"\n"+
        'Email: gaztaroa@gaztaroa.com'
    }
    ];
```

A continuación, se crea el fichero ContactoComponent.js que se guarda en la carpeta componentes. Se realiza un componente funcional llamado *RenderContacto*, el cual utiliza el elemento de react native Card. La información la obtiene gracias a un props.

```
function RenderContacto(props) {

    const contacto = props.contacto;

        if (contacto != null) {
            return(
            <Card>
              <Card.Title>{contacto.nombre}</Card.Title>
              <Card.Divider/>
              <Text style={{margin: 20}}>
                {contacto.descripcion}
              </Text>
            </Card>
            );
        }
        else {
            return(<View></View>);
        }
}
```

En la clase Contacto se llama a *RenderContacto* y se le pasa el estado de contactos con id=0, que es la única que existe.

```
class Contacto extends Component {
  constructor(props) {
      super(props);
      this.state = {
          contactos: CONTACTOS
      };
  }

  render(){
      return (<RenderContacto contacto={this.state.contactos[0]} /> );
  }
}

export default Contacto;
```

Finalmente se realizan los cambios en *CampoBaseComponent.js*, creando el componente funcional *ContactoNavegador*. Este componente llama *ContactoComponent.js.*

```
function ContactoNavegador() {
  return (
    <Stack.Navigator
      initialRouteName="Contacto"
      headerMode="screen"
      screenOptions={{
        headerTintColor: '#fff',
        headerStyle: { backgroundColor: '#015afc' },
        headerTitleStyle: { color: '#fff' },
      }}>

      <Stack.Screen
        name="Contacto"
        component={Contacto}
        options={{
          title: 'Contacto',
        }}
      />
    </Stack.Navigator>
  );
}
```

También será necesario añadir al componente funcional *DrawerNavegador*, el *ContactoNavegador.*

```
<Drawer.Screen name="Contacto" component={ContactoNavegador} />
```

Como resultado final se obtiene lo siguiente:

2. Componente Quiénes Somos

En este componente se va a tener una *Card* con la historia del club y otra *Card* con varios *ListItem*s, embebidos en un *FlatList*, con información de las actividades y recursos de Gaztaroa.

Al igual que con Contacto se crea en la carpeta común un fichero con una constante a la cual se le asigna un array en el que se guarda la historia del grupo Gaztaroa.

A continuación, se crea en la carpeta componentes el fichero *QuienesSomosComponent.js* se crea un componente funcional llamado Historia donde se crea una *Card* que contendrá la información que se le pasará.

```
function Historia(props) {

    const historia = props.historia;

        if (historia != null) {
            return(
            <Card>
              <Card.Title>{historia.nombre}</Card.Title>
              <Card.Divider/>
              <Text style={{margin: 20}}>
                {historia.descripcion}
              </Text>
            </Card>
            );
        }
        else {
            return(<View></View>);
        }
}
```

Gracias al componente *renderQuienesSomosItem* se creará una lista parecida a la del *Calendario*.

```
const renderQuienesSomosItem = ({item, index}) => {
    return (
        <ListItem  key={index}  bottomDivider>
            <Avatar source={require('./imagenes/40Años.png')} />
            <ListItem.Content>
                <ListItem.Title>{item.nombre}</ListItem.Title>
                <ListItem.Subtitle>{item.descripcion}</ListItem.Subtitle>
            </ListItem.Content>
        </ListItem>
    );
};
```

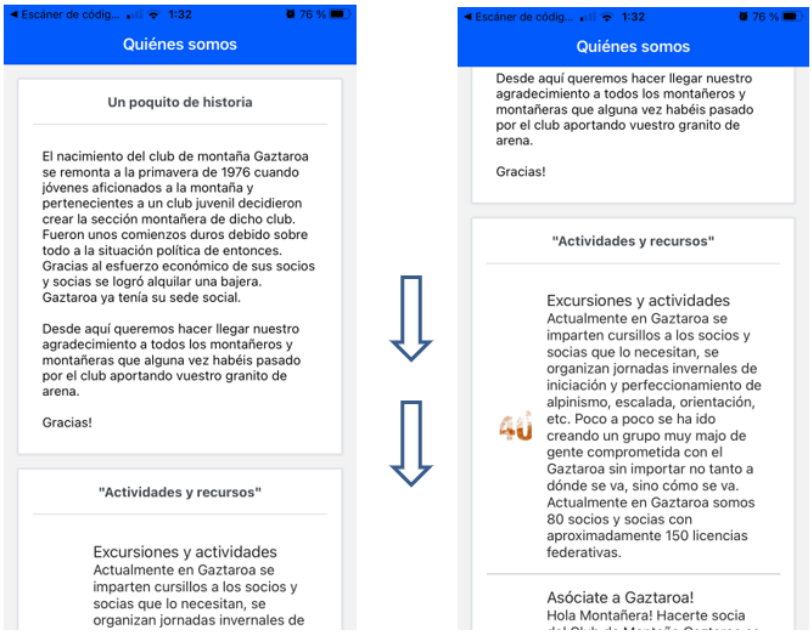
En el return se llamará al componente funcional Historia y se le enviará el estado de historia, al igual que se ha hecho con Contacto. El *FlatList* se encontrará dentro de un *Card* y haciendo uso de *renderQuienesSomosItem* y obteniendo como datos el estado de actividades se obtendrá la lista deseada. Todo esto se encontrará dentro de un *ScrollView* para poder hacer scroll.

```
return (
    <ScrollView>
        <Historia historia={this.state.historia[0]} />

        <Card>
          <Card.Title>"Actividades y recursos"</Card.Title>
          <Card.Divider/>
          <SafeAreaView>
        <FlatList
            data={this.state.actividades}
            renderItem={renderQuienesSomosItem}
            keyExtractor={item => item.id.toString()}
        />
        </SafeAreaView>
        </Card>
    </ScrollView>
```

Para terminar, se realizan los cambios en *CampoBaseComponent.js*, creando el componente funcional *QuienesSomosNavegador.* Este componente llama *QuienesSomosComponent.js.* Será necesario añadir al componente funcional *DrawerNavegador*, el *QuienesSomosNavegador,* al igual que se hizo con *Contacto*.

Como resultado final se obtiene:



3. Componente Home

Por último, actualizaremos el componente *Home* para que su título se muestre por encima de la imagen, ajustando además su color a *chocolate* (color: 'chocolate'), empleando para ello un *StyleSheet*.

Para conseguir esto basta con modificar el componente funcional *RenderItem*, donde el Card.Title se introduce dentro de Card.Image y se le asigna como color 'chocolate' y para aumentar el tamaño de la fuente fontSize:30.

```
function RenderItem(props) {

        const item = props.item;

        if (item != null) {
            return(
                <Card>
                    <Card.Divider/>
                    <Card.Image source={require('./imagenes/40Años.png')}>
                    <Card.Title style={{color:'chocolate', fontSize:30}}>{item.nombre}</Card.Title>
                    </Card.Image>
                    <Text style={{margin: 20}}>
                        {item.descripcion}
                    </Text>
                </Card>
            );
```

El resultado final es el siguiente: